

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

The following is a complete listing of all claims in the application, with an indication of the status of each:

Listing of claims:

1 1. (previously presented) An automated method for converting tables of data
2 in a source database to components of a target application, the method
3 comprising the steps:
4 obtaining by a computer identification of a) specified tables in said
5 source database containing data usable in said target application, b) a target
6 location for said target application and c) an application server being used for
7 development of said target application at said target location
8 reading by a computer definitions of said specified tables from said
9 source database; and
10 generating by a computer from said database definitions a plurality of
11 source code files in a language of said target application, each said specified
12 table being referenced consistently across said plurality of source code files,
13 said plurality of source code files including object classes and deployment
14 descriptors, said specified tables being made accessible to a remote client by
15 said target application, said target application being developed using said
16 plurality of source code files.

1 2. (previously presented) The method of claim 1 further comprising the steps
2 of:
3 providing a user interface to permit interaction with the computer
4 application program;
5 including in said obtaining step identification of d) an Enterprise
6 JAVA Bean Java Archive (EJB Jar) file;

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

7 generating said plurality of source code files in said identified target
8 location, said identified target location being a user-specified directory path,
9 said application server supporting development of Enterprise JAVA Beans
10 (EJBs) and supporting a Java Naming and Directory Interface (JNDI) to assist
11 with said consistent table references, and said plurality of source code files
12 comprising

13 a Data JAVA file for each specified table, a Home Interface JAVA file
14 for each specified table, a Remote Interface JAVA file for each specified table,
15 a Bean JAVA file for each specified table, a Primary Key JAVA file for each
16 specified table, a Persistent JAVA file for each specified table, an Enterprise
17 JAVA Bean Deployment Descriptor XML file, an Enterprise JAVA Bean Jar
18 batch command file, a Vendor-Specific Deployment XML file, and a Vendor-
19 Specific Build batch command file.

1 3. (previously presented) The method of claim 2, wherein providing a user
2 interface further comprises the steps of:

3 querying the database to get names of all tables related to the database,
4 from which each of said specified tables are selected for encapsulation with an
5 EJB; and

6 querying the database to acquire information about fields within each
7 of said specified tables.

1 4. (previously presented) The method of claim 2, wherein generating the
2 Data JAVA file further comprises the steps of:

3 generating a file with its filename equal to a table name followed by
4 the word "Data" and file extension ".java";

5 writing header comments indicating that a JAVA class encapsulates
6 one row of the specified table;

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

7 writing a JAVA package definition consistent with the user-specified
8 directory path;
9 writing a first JAVA statement to import a java.io.Serializable class;
10 writing a JAVA class definition with a class name equal to its
11 filename;
12 writing a second JAVA statement indicating that the class implements
13 the Serializable interface;
14 writing a corresponding attribute definition for each one of a plurality
15 of fields in the specified table;
16 ensuring that a JAVA data type of each one of a plurality of attributes
17 is compatible with a database data type of its corresponding field;
18 ensuring that a name of each one of the plurality of attributes is
19 identical to the name of its corresponding field;
20 writing a first JAVA public method to get a value of each one of the
21 plurality of attributes;
22 writing a second JAVA public method to set the value of each one of
23 the plurality of attributes; and
24 writing a third JAVA public method named "toString" to return a
25 string representing a Data class.

1 5. (previously presented) The method of claim 2, wherein generating the
2 Home Interface JAVA file further comprises the steps of:
3 generating a file with filename equal to a table name followed by the
4 word "Home" and file extension "java";
5 writing header comments indicating that a JAVA class is the Home
6 Interface for the Bean encapsulating the specified table;
7 writing a JAVA package definition consistent with the user-specified
8 directory path;

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

9 writing a first JAVA statement to import the following classes:
10 javax.ejb.EJBHome; javax.ejb.FinderException; javax.ejb.CreateException;
11 java.rmi.RemoteException; java.util.Collection;
12 writing a JAVA interface definition with an interface name equal to
13 the filename;
14 writing a second JAVA statement indicating that the class extends the
15 EJBHome interface;
16 writing a first JAVA method signature named "create" which takes as
17 parameters the attributes of Data class and returns an object of Remote class;
18 writing a second JAVA method signature named "create" which takes
19 as a parameter an object of Data class and returns an object of Remote class;
20 and
21 writing a third JAVA method signature named "findByPrimaryKey"
22 which takes as a parameter an object of PrimaryKey class and returns an
23 object of Remote class wherein writing the JAVA method signature named
24 "findByPrimaryKey" has one primary key and the computer application
25 program makes the parameter its corresponding attribute of Data class.

1 6. (previously presented) The method of claim 2, wherein generating the
2 Remote Interface JAVA file further comprises the steps of:
3 generating a file with its filename equal to the table name followed by
4 the word "Remote" and file extension ".java";
5 writing header comments indicating that this JAVA class is the
6 Remote interface for the Bean encapsulating the specified table;
7 writing a JAVA package definition consistent with the user-specified
8 directory path;
9 writing a first JAVA statement to import the following classes:
10 javax.ejb.EJBObject; java.rmi.RemoteException;

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

11 writing a JAVA interface definition with its interface name equal to
12 the filename;
13 writing a second JAVA statement indicating that the class extends the
14 EJBObject interface;
15 writing a first JAVA public method signature to get a value of each
16 attribute and wherein the signature indicates that the method may throw a
17 Remote Exception; and
18 writing a second JAVA public method to set the value of each attribute
19 signature and wherein the signature shall indicate that the method may throw a
20 Remote Exception.

1 7. (previously presented) The method of claim 2, wherein generating the
2 Bean JAVA file further comprises the steps of:
3 generating a file with its filename equal to its table name followed by
4 the word "Bean" and file extension "java";
5 writing header comments indicating that the JAVA class is an EJB
6 which encapsulates one row of the specified table;
7 writing a JAVA package definition consistent with the user-specified
8 directory path;
9 writing a first JAVA statement to import the following classes:
10 java.rmi.RemoteException; all classes in the javax.ejb package; all classes in
11 the java.util package;
12 writing a JAVA class definition with its class name equal to the
13 filename;
14 writing a second JAVA statement indicating that the class implements
15 the EntityBean interface;
16 writing an attribute definition for one attribute named "theContext" of
17 type "EntityContext";

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

18 writing, for each field in the specified table, its corresponding attribute
19 definition;
20 ensuring that the JAVA data type of each attribute is compatible with
21 the database data type of its corresponding field;
22 ensuring that each attribute is identical to the lower-case name of its
23 corresponding field;
24 writing a first JAVA public method to get the value of each attribute;
25 writing a second JAVA public method to set the value of each
26 attribute;
27 writing a third JAVA public method named "setAll" which has an
28 object of Data class as parameter and can throw a Remote Exception and
29 wherein each attribute of Bean class is set to the value of its corresponding
30 Data attribute;
31 writing a fourth JAVA public method named "getAll" which returns an
32 object of Data class and can throw a Remote Exception;
33 writing a fifth JAVA public method named "ejbCreate" with attributes
34 of Data class as parameters and a signature of this method says it returns an
35 object of type "String" but the body of the method must return "null" and this
36 method can throw a Create Exception and a Remote Exception;
37 writing a sixth JAVA public method named "ejbPostCreate" with
38 attributes of Data class as parameters and the signature of this method says it
39 returns an object of type "String" but the body of the method must return
40 "null" and this method can throw a Create Exception and a Remote Exception;
41 writing a seventh JAVA public method named "ejbPostCreate" with an
42 object of Data class as the parameter and the signature of this method says it
43 returns an object of type "String" but the body of the method must return
44 "null" and this method can throw a Create Exception and a Remote Exception;

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

45 writing an eighth JAVA public method named "ejbLoad" which can
46 throw an EJB Exception or a Remote Exception and the body of this method
47 may be empty;
48 writing a ninth JAVA public method named "ejbStore" which can
49 throw an EJB exception or a Remote exception and the body of this method
50 may be empty;
51 writing a tenth JAVA public method named "ejbActivate" which can
52 throw an EJB Exception or a Remote Exception and the body of this method
53 can be empty;
54 writing an eleventh JAVA public method named "ejbPassivate" which
55 can throw an EJB Exception or a Remote Exception and the body of this
56 method can be empty;
57 writing a twelfth JAVA public method named "ejbRemove" which can
58 throw an EJB Exception or a Remote Exception and the body of this method
59 can be empty;
60 writing a thirteenth JAVA public method named "setEntityContext"
61 which can throw an EJB exception or a Remote Exception and the parameter
62 for this method is an object of type EntityContext and is used to set
63 theContext;
64 writing a fourteenth JAVA public method named "unsetEntityContext"
65 which can throw an EJB Exception or a Remote Exception and this method
66 sets theContext to "null"; and
67 writing a fifteenth JAVA public method named "toString" to return a
68 string representing the Bean class.

1 8. (previously presented) The method of claim 2, wherein generating the
2 Primary Key JAVA file further comprises the steps of:

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

3 generating a file with at least two primary key fields such that its
4 filename is made equal to its table name followed by the word "PrimaryKey"
5 and file extension ".java";
6 writing header comments indicating that this JAVA class encapsulates
7 the Primary Key of the specified table;
8 writing a JAVA package definition consistent with the user-specified
9 directory path;
10 writing a first JAVA statement to import the java.io.Serializable class;
11 writing a JAVA class definition with the class name equal to its
12 filename;
13 writing a second JAVA statement indicating that its class implements
14 the Serializable interface;
15 writing a corresponding attribute definition for each primary key field;
16 ensuring that the JAVA data type of each attribute is compatible with
17 the database data type of its corresponding field;
18 ensuring that each attribute is identical to the lower case name of its
19 corresponding field;
20 writing a first JAVA public method to get the value of each attribute;
21 writing a second JAVA public method to set the value of each
22 attribute;
23 writing a third JAVA public method named "equals" which returns a
24 boolean value and the parameter to this method is an object of the Object class
25 and returns "true" if the parameter is an instance of the Primary Key class and
26 if the parameter's attributes have values equal to the values of the attributes of
27 this object; and
28 writing a fourth JAVA public method named "hashCode" which returns
29 an integer value and this method forms a String of the attribute values and
30 returns a hash code of that String.

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

1 9. (previously presented) The method of claim 2, wherein generating the
2 Persistent JAVA file further comprises the steps of:
3 generating a file with filename equal to the table name followed by the
4 word "Persistent" and file extension "java";
5 writing header comments indicating that this JAVA class encapsulates
6 one row of the specified table and this client side class can be passed to the
7 server for execution;
8 writing a JAVA package definition consistent with the user specified
9 directory path;
10 writing a first JAVA statement to import the following classes
11 javax.naming.InitialContext; javax.naming.NamingException;
12 javax.rmi.PortableRemoteObject; java.sql.Connection; java.io.Serializable; all
13 classes of the utility package com.lmco.util;
14 writing a JAVA class definition with the class name equal to the
15 filename;
16 writing a second JAVA statement indicating that the class extends its
17 corresponding Data class;
18 writing a third JAVA statement indicating that the class implements
19 the Serializable interface and the PersistentObject interface;
20 writing a first JAVA public method to construct an object of this class
21 without any parameters;
22 writing a second JAVA public method to construct an object of this
23 class with the attributes of the Data class as parameters;
24 writing a third JAVA public method to construct an object of this class
25 with an object of the Data class as parameters;
26 writing a fourth JAVA public method named "create" which returns a
27 boolean and the body of this method provides a JAVA try-and-catch block to

28 invoke the "create" method of the Home interface with the attributes of this
29 class as parameters and if an exception occurs, the result is false or otherwise
30 the result is true;

31 writing a fifth JAVA public method named "read" which returns a
32 boolean and the body of this method provides a JAVA try-and-catch block to
33 invoke the "findByPrimaryKey" method of the Home interface with the
34 Primary Key class as parameter and if an exception occurs, the result is false
35 or otherwise, the result is true and the attributes of this class are set to the
36 values in the Remote interface returned by the "findByPrimaryKey" method;

37 writing a sixth JAVA public method named "update" which returns a
38 boolean and the body of this method provides a JAVA try-and-catch block to
39 invoke the "findByPrimaryKey" method of the Home interface, with the
40 primary key class as parameter and if an exception occurs, the result is false or
41 otherwise, the result is true and the attributes of the Remote interface are set to
42 the values of the attributes in this class;

43 writing an seventh JAVA public method named "delete" which returns
44 a boolean and the body of this method provides a JAVA try-and-catch block to
45 invoke the "remove" method of the Home interface with the Primary Key class
46 as parameter and if an exception occurs, the result is false or otherwise the
47 result is true; and

48 writing a JAVA protected method named "getHome" to return an
49 object of the Home class and the body of this method providing a JAVA try-
50 and-catch block to use an object of type InitialContext to look up in the JNDI
51 a name, which JNDI name is equal to the name of the specified table.

1 10. (previously presented) The method of claim 2, wherein generating the
2 EJB Deployment Descriptor XML file further comprises the steps of:

FS-00534

09/916,516

02890034aa

Amendment dated 06/21/2005

Reply to office action mailed 04/21/2005

3 generating a file with filename "ejb-jar.xml" and located in a folder
4 named META-INF one directory deep within the user-specified target
5 directory path;
6 writing an XML header statement for documents of type "ejb-jar";
7 writing a first tag <ejb> to begin the document;
8 writing a second tag <description> to begin a description;
9 writing the description of a jar file which includes the package name
10 and the identified EJB jar filename;
11 writing a third tag </description> to end the description;
12 writing a fourth tag <enterprise-beans> to begin a list of EJB's;
13 writing a fifth tag <entity> to begin a definition of its corresponding
14 entity EJB;
15 writing a sixth tag <description> to begin the description of the EJB;
16 writing a description of the EJB including the name of its
17 corresponding specified table;
18 writing a seventh tag </description> to end the description;
19 writing an eighth tag <ejb-name> to begin the name of the EJB;
20 writing the name of its corresponding specified table;
21 writing a ninth tag </ejb-name> to end the name of the EJB;
22 writing a tenth tag <home> to begin a home of the EJB;
23 writing a fully-qualified-name of its corresponding Home class;
24 writing an eleventh tag </home> to end the home of the EJB;
25 writing a twelfth tag <remote> to begin a remote interface of the EJB;
26 writing a fully-qualified-name of its corresponding Remote class;
27 writing a thirteenth tag </remote> to end the remote of the EJB;
28 writing a fourteenth tag <ejb-class> to begin a bean class of the EJB;
29 writing the fully-qualified-name of its corresponding Bean class;
30 writing a fifteenth tag </ejb-class> to end the bean class of the EJB;

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

31 specifying container-maintained persistence with the statement
32 <persistence-type>Container</persistence-type>;
33 writing a sixteenth tag <prim-key-class> to begin the primary key class
34 of the EJB;
35 writing the fully-qualified-name of its corresponding PrimaryKey
36 class;
37 writing a seventeenth tag </prim-key-class> to end the primary key
38 class of the EJB;
39 specifying the EJB as not re-entrant by writing
40 <reentrant>False</reentrant>;
41 beginning each field of the specified table with <cmp-field><field-
42 name>;
43 writing each field of the specified table;
44 ending each field of the specified table with </field name></cmp-
45 field>;
46 if the specified table has only one primary key field, writing the
47 primary key field as <primkey-field> followed by the field name followed by
48 </primkey-field>;
49 writing an eighteenth tag </entity> to end a description of its
50 corresponding entity EJB for each specified table;
51 writing a nineteenth tag <assembly-descriptor> to begin the assembly
52 descriptor;
53 writing a twentieth or more tags defining a default security role;
54 providing default permission to all methods of all EJB's;
55 specifying container-managed transactions for all methods of each
56 EJB;
57 writing a twenty-first or greater tag <assembly-descriptor> to end
58 assembly;

FS-00534
Amendment dated 06/21/2005

09/916,516

02890034aa
Reply to office action mailed 04/21/2005

59 writing a twenty-second or greater tag `</enterprise-beans>` to end the
60 list of EJB's; and
61 writing a twenty-third or greater tag `</ejb>` to end the document.

1 11. (previously presented) The method of claim 2, wherein generating the
2 EJB Jar batch command file further comprises the steps of:
3 generating a file with filename set to the identified name of the Jar file
4 where the file extension on MS Windows-based systems is "bat";
5 writing a command to compile each generated Java file;
6 writing a command to put the ejb-jar.xml file into one Jar file; and
7 for each specified table, writing a command to put all of generated
8 JAVA classes into the Jar file.

1 12. (previously presented) The method of claim 2, wherein generating the
2 Vendor-Specific Deployment XML file further comprises the steps of:
3 generating a file with filename indicating both the vendor and the
4 identified name of the EJB Jar file;
5 generating one or more XML tags according to vendor specifications
6 for a target Application Server and for which the details will vary with the
7 Vendor;
8 generating one or more XML tags to specify a JNDI name for the EJB
9 corresponding to each specified table and the JNDI name is the same as its
10 specified table name; and
11 generating one or more XML tags to specify mapping of each EJB
12 attribute to its field in the specified database table.

1 13. (previously presented) The method of claim 2, wherein generating the
2 Vendor-Specific Build batch command file further comprises the steps of:

FS-00534

09/916,516

02890034aa

Amendment dated 06/21/2005

Reply to office action mailed 04/21/2005

- 3 generating a file with its filename indicating both a vendor and the
- 4 word "command" and wherein the file extension on MS Windows-based
- 5 systems is "bat";
- 6 generating vendor-specific commands to import and deploy a Jar file
- 7 according to the Vendor-Specific Deployment file.